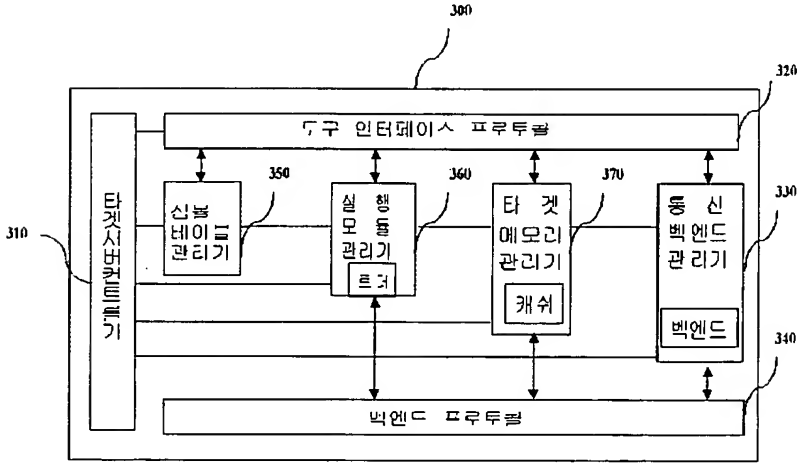
KOREAN PATENT ABSTRACTS(KR)

Document Code:A
(11) Publication No.1020010038541 (43) Publication.Date. 20010515
(21) Application No.1019990046545 (22) Application Date. 19991026
(51) IPC Code:
G06F 15/16

(71) Applicant:
KOREA ELECTRONICS & TELECOMMUNICATIONS RESEARCH INSTITUTE
(72) Inventor:
KIM, HEUNG NAM
LEE, U JIN
LIM, CHAE DEOK
OH, GIL ROK
(30) Priority:

(54) Title of Invention
TARGET SERVER AND CONTROL METHOD FOR REMOTELY DEVELOPING EMBEDDED REAL TIME SOFTWARE

Representative drawing



(57) Abstract:

PURPOSE: A target server and a control method for remotely developing an embedded real time software is provided to allow a target server to be responsible for a communication between a tool and a host system, a symbol table management, a target memory management, and an execution file management so that it can reduce resources of the target system.

CONSTITUTION: The system comprises a target server controller(310), a symbol table manager(350), an execution module manager(360), a communication back end manager(330), and a target memory manager(370). The target server controller(370)

analyzes the functions requested by a tool, and calls the corresponding function. The symbol table manager(350) manages the data on subroutines, variables and module identifications included in the all the execution modules loaded in a target system and a system symbol table. The execution module manager(360) loads or unloads the execution file at or from the target system, and manages the loaded module list on the host system.

10-2001-0038541

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(51) Int. Cl.[7]
G06F 15/16

(11) 공개번호 10-2001-0038541
(43) 공개일자 2001년05월15일

(21) 출원번호 10-1999-0046545
(22) 출원일자 1999년10월26일

(71) 출원인 한국전자통신연구원 정선종
대전 유성구 가정동 161번지
(72) 발명자 임채덕
대전광역시유성구전민동엑스포아파트306-1203
이우진
대전광역시서구만년동281상록수아파트105-1307
김흥남
대전광역시유성구도룡동383-2과기원교수아파트2-301
오길록
서울특별시강남구삼성동19-4상아아파트3-302
(74) 대리인 전영일

심사청구 : 있음

(54) 내장형 실시간 소프트웨어의 원격 개발을 위한 타겟 서버 장치 및 그 제어 방법

요약

본 발명은 내장형 실시간 소프트웨어를 호스트 상의 여러 개발 도구를 이용하여 원격 개발할 수 있도록 해 주는 타겟 서버 장치 및 그 제어 방법을 제공하는데 그 목적이 있다.

본 발명에 따르면, 기 설정한 플래그(Flag)를 반영하여 그 특성에 맞게 상기 타겟 서버를 구동시키고, 상기 호스트 시스템 상의 도구가 요청한 기능을 해석하여 거기에 맞는 서비스를 호출해 주는 기능을 수행하는 타겟 서버 컨트롤기와; 상기 타겟 서버 컨트롤기에 의하여 호출된 서비스에 따라, 상기 타겟 시스템의 시스템 심볼 테이블과 상기 타겟 시스템에 로딩된 모든 실행 모듈들의 서브루틴, 변수 및 모듈 id에 대한 정보를 관리하는 기능을 수행하는 심볼 테이블 관리기와; 상기 호스트 시스템에 있는 실행 파일을 타겟 시스템에 로딩/언로딩하는 기능, 로딩된 모듈 리스트를 상기 호스트 시스템 상에서 관리하는 기능을 수행하는 실행 모듈 관리기와; 타겟 서버 구동시 상기 호스트 시스템 상의 도구들과 상기 타겟 시스템이 통신할 수 있도록 하는 통신 백엔드 관리기를 포함하여 이루어진 것을 특징으로 하는 타겟 서버 제어 장치가 제공된다.

대표도

도3

명세서

도면의 간단한 설명

도 1은 종래 기술에 따른 내장형 소프트웨어의 원격 개발 도구와 타겟 시스템의 연결 구조도이고,

도 2는 본 발명에 일 실시예에 따른 내장형 소프트웨어의 원격 개발 도구와 타겟 시스템의 연결 구조도이고,

도 3은 본 발명의 일 실시예에 따른 내장형 소프트웨어의 원격 개발 시스템의 구성도이고,

도 4는 본 발명의 일 실시예에 따른 타겟 서버에 대한 구조도이고,

도 5는 도 4에 도시된 도구 인터페이스 프로토콜의 기능에 따른 분류 테이블이고,

도 6은 본 발명의 일 실시예에 따른 타겟 서버 장치가 도구에 대한 서비스를 제공하기 위하여 서비스 데몬으로 동작되는 과정을 나타내는 흐름도이고,

도 7은 본 발명의 일 실시예에 따른 타겟 서버 장치가 데몬으로 작동하고 있으면서, 호스트 상의 여러 개발 도구와 타겟 에이전트의 서비스 요청을 받아들여 서비스해주는 과정을 나타낸 흐름도이고,

도 8은 본 발명의 일 실시예에 따른 호스트 시스템에서 교차 컴파일(Cross Compile)한 내장형 소프트웨어

를 타겟 서버 장치를 이용하여 타겟에 로딩하여 실행하는 과정을 나타낸 흐름도이다.

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 내장형 실시간 소프트웨어를 호스트 상의 여러 개발 도구를 이용하여 원격 개발할 수 있도록 해 주는 타겟 서버 장치 및 그 제어 방법에 관한 것으로, 특히, 호스트 시스템 상에 여러 도구를 사용하여 원격 개발을 할 수 있는 환경에서 타겟 시스템에 접근하기 위하여 하나의 창구를 통해 모든 개발 행위가 수행될 수 있도록 지원하는 도구 중개자의 역할을 수행하는 타겟 서버 장치 및 그 제어 방법에 관한 것이다.

정보 가전에 탑재할 내장 시스템은 작은 컴퓨터라고 볼 수 있지만, 우리가 보통 사용하는 개인용 컴퓨터와는 비교가 안될 정도로 작은 규모이다. 그래서 디버거와 같이 리소스를 많이 필요로 하는 대형 소프트웨어를 내장 시스템에 탑재하여 응용 프로그램을 개발하기는 상당히 어렵다. 그래서, 타겟 시스템과 시리얼 혹은 이더넷으로 연결하여 원격 컴퓨터(이하 호스트 시스템이라 칭함)에서 내장 시스템에 탑재할 응용을 개발하려는 시도가 있어 왔다.

종래에는 호스트 상에 디버거 클라이언트를 두고 타겟에 디버거 서버를 두는 GDB(Gnu DeBugger)와 같은 디버거 하나 만으로 응용을 개발하는 게 대부분의 내장형 소프트웨어의 개발 환경이었으나, 최근 다양한 응용을 지원하고, 응용 개발의 효율을 기하기 위해 디버거 외에도 원격 쉘, 타겟 자원 모니터 등 다양한 도구를 호스트 상에 둘 필요성이 제기되었다.

내장형 프로세서인 32 비트 마이크로 프로세서가 값이 싸지고, 특정 응용 전용인 내장형 마이크로 프로세서의 출현으로 말미암아 통신 단말기 및 디지털 TV와 같은 정보 가전 제품에 탑재할 내장형 실시간 응용의 수요가 급증하고 있다. 더구나 내장형 응용이 멀티미디어와 같은 대형 데이터를 처리해야 하는 등 점점 복잡해지고 있으며, 이를 위한 개발 환경은 타겟 시스템(Target System)의 자원이 매우 제한적이라서, 호스트 시스템(Host System)과 타겟 시스템 간의 통신 부담이 크다는 문제점이 있다.

내장형 소프트웨어를 원격지의 호스트 시스템 상에서 개발하기 위해서 종래에는 호스트 시스템 상에 디버거 클라이언트(Debugger Client)를 두고 타겟 시스템에 디버거 서버(Debugger Server)를 둔 GDB와 같은 원격 디버거를 사용하였으나, 내장형 소프트웨어가 점점 복잡해지고, 원격 개발 환경이 타겟 시스템에 독립적이며 다양한 개발 도구를 사용하면서 실시간으로 타겟 시스템의 상황을 파악할 필요성이 제기되었다. 이러한 필요성을 만족시키기 위해서 호스트 시스템 상에 여러 도구를 두고 타겟 시스템에 연결하여 내장형 소프트웨어 개발하는 환경을 구축하기 시작했다.

도 1은 종래 기술에 따른 내장형 소프트웨어의 원격 개발 도구와 타겟 시스템의 연결 구조도로서, 이를 상세히 설명하면 다음과 같다.

도 1에 나타난 바와 같이 개발자 도구가 여러 개 있을 경우, 두 가지 경우를 고려할 수 있다. 기존의 서로 다른 공급자(Vendor)의 상용화 도구를 사용하여 개발 환경을 꾸밀 경우, 가능한 호스트 시스템과 타겟 시스템 간의 연결 구조는 도 1의 110과 같다. 그러나, 도 1의 110 구조는 각각의 도구가 타겟 시스템과 연결하여 타겟 시스템의 자원에 접근하는데, 이 경우는 각 도구가 동시에 접근할 경우, 타겟 시스템의 통신 부담이 매우 커진다. 그리고, 도구를 사용하는 응용 개발자가 타겟 시스템에 심각한 오류를 발생시켰을 경우에, 다른 도구를 사용하는 개발자도 타겟 시스템을 사용할 수 없게 된다. 서로 다른 도구일지라도 타겟 시스템의 심볼 테이블(Symbol Table)과 같이 중복된 데이터를 각 도구가 가지고 있어야 하고, 경우에 따라서는 중복 데이터에 대한 일관성도 문제가 될 수 있다. 또한, 호스트 시스템 상에 새로운 도구를 추가할 경우에도 통신 방식, 타겟 시스템에 대해 상세히 파악해야 하므로 도구 개발자에게 큰 부담을 줄 수 있다는 문제점이 있다.

'이은향 외 2인'이 등록받은 선행 특허 '교차 디버깅 서버에서 프로세스 실행 제어 서비스 실현 방법'(등록 번호 : 1997-069487)은, 타겟 시스템에서 동작하는 교차 디버깅 서버에서 프로세스 실행 제어 서비스 실현 방법에 관한 것이다.

상기 선행 특허에서 제공하는 교차 디버깅 서버에서 프로세스 실행 제어 서비스를 실현하는 방법은 호스트 시스템에 있는 교차 디버깅 클라이언트로부터 프로세스 실행 제어 서비스 요청을 수락하여, 그 서비스가 어떠한 형태의 서비스 요청인가를 판단하는 제 1 단계, 상기 제 1 단계에서 요청받은 프로세스 실행 제어 서비스의 종류에 따라 해당 서비스를 수행하는 제 2 단계, 상기 제 2 단계의 수행 결과를 호스트 시스템에 있는 교차 디버깅 클라이언트에게 제공하는 제 3 단계 등 세 단계로 구성된다.

상기 프로세스 실행 제어 서비스를 제공하기 위한 실시간 운영 체제의 교차 디버깅 서버는 하나 이상의 서비스 데몬, 디버깅 서버 엔진 및 원격 통신 처리부로 이루어져 있다. 상기 서비스 데몬은 교차 디버깅 클라이언트로부터 실행 제어 서비스를 요청받아 디버깅 서비스 엔진을 이용하여 사용자 프로그램 내의 임의의 프로세스를 실행 중단 / 재개하는 등의 프로세스 실행을 제어하며 그 결과를 원격 통신 처리부를 통하여 교차 디버깅 클라이언트에게 전달하는 역할을 한다. 프로세스 실행 중단 요청을 받으면, 해당 프로세스를 찾아 프로세스의 현재 상태를 저장하고, 실행 재개할 때의 프로세스의 상태를 정의한다.

만약 실행 중단시킬 프로세스가 어떠한 사건이 일어나기를 기다리는 대기 상태가 아니면 프로세스 컨텍스트(Context) 스위치에 필요한 정보를 프로세스 실행 제어 블록에 저장하고, 프로세스 스케쥴링 준비 큐에서 제외한다.

만약 해당 프로세스가 대기 상태이면, 대기 상태로 만든 시스템 콜의 시작 지점을 프로세스 스택을 찾아

가며 찾아서, 프로그램 카운터가 그 시작 지점을 가리키도록 조정하고, 스택 포인터와 프레임 포인터가 시스템 콜의 시작 지점에 해당하는 스택 프레임을 가리키도록 조정한다. 마지막으로 상기 시스템 콜이 사용한 입력 파라미터들을 스택 프레임에서 찾아 파라미터 전달용 레지스터에 재로딩하여 놓고 나머지 부분은 대기 상태가 아닌 경우의 처리를 수행한다. 이렇게 함으로써, 시스템 실행이 중단되는 상황의 발생이 없이 실행 중인 프로세스만을 실행 중단시키고, 실행 재개시키는 과정을 제공하여 실행 중인 프로세스를 디버깅함에 있어서 프로세스의 실행 제어를 쉽게 할 수 있을 뿐만 아니라, 디버깅에 따르는 일련의 과정을 단축할 수 있어서 효율적인 면을 제시하였다.

그러나, 본 선행 특허에서도 위에서 서술한 문제점을 극복하고 있지 아니하다.

발명이 이루고자하는 기술적 과제

본 발명은 상기와 같은 종래기술의 문제점을 해결하기 위하여 안출된 것으로서, 호스트 시스템 상에 여러 도구를 사용하여 원격 개발을 할 수 있는 환경에서 타겟 시스템에 접근하기 위하여 하나의 창구를 통해 모든 개발 행위가 수행될 수 있도록 지원하는 타겟 서버 장치 및 그 제어 방법을 제공하는데 그 목적이 있다.

발명의 구성 및 작용

앞서 설명한 바와 같은 목적을 달성하기 위한 본 발명에 따르면, 타겟 시스템(Target System)에서 운영되는 내장형 실시간 소프트웨어(Embedded Real-Time Software)를 호스트 시스템(Host System) 상의 개발 도구를 이용하여 원격 개발할 수 있도록 하는 타겟 서버(Target Server) 장치에 있어서, 기 설정한 플래그(Flag)를 반영하여 그 특성에 맞게 상기 타겟 서버를 구동시키고, 상기 호스트 시스템 상의 도구가 요청한 기능을 해석하여 거기에 맞는 서비스를 호출해 주는 기능을 수행하는 타겟 서버 컨트롤기와; 상기 타겟 서버 컨트롤기에 의하여 호출된 서비스에 따라, 상기 타겟 시스템의 시스템 심볼 테이블과 상기 타겟 시스템에 로딩된 모든 실행 모듈들의 서브루틴, 변수 및 모듈 id에 대한 정보를 관리하는 기능을 수행하는 심볼 테이블 관리기와; 상기 호스트 시스템에 있는 실행 파일을 타겟 시스템에 로딩/언로딩하는 기능, 로딩된 모듈 리스트를 상기 호스트 시스템 상에서 관리하는 기능을 수행하는 실행 모듈 관리기와; 타겟 서버 구동시 상기 호스트 시스템 상의 도구들과 상기 타겟 시스템이 통신할 수 있도록 하는 통신 백엔드 관리기를 포함하여 이루어진 것을 특징으로 하는 타겟 서버 제어 장치가 제공된다.

또한, 타겟 시스템(Target System)에서 운영되는 내장형 실시간 소프트웨어(Embedded Real-Time Software)를 호스트 시스템(Host System) 상의 개발 도구를 이용하여 원격 개발할 수 있도록 하는 타겟 서버(Target Server) 제어 방법에 있어서, 기 설정한 플래그(Flag)를 반영하여 그 특성에 맞게 상기 타겟 서버를 구동시키고, 상기 호스트 시스템 상의 도구가 요청한 기능을 해석하여 거기에 맞는 서비스를 호출해 주는 제 1 단계와; 상기 제 1 단계에서 호출된 서비스에 따라, 상기 타겟 시스템의 시스템 심볼 테이블과 상기 타겟 시스템에 로딩된 모든 실행 모듈들의 서브루틴, 변수 및 모듈 id에 대한 정보를 관리하며 상기 호스트 시스템에 있는 실행 파일을 상기 타겟 시스템에 로딩/언로딩한 후, 로딩된 모듈 리스트를 상기 호스트 시스템 상에서 관리하는 제 2 단계를 포함하여 이루어진 것을 특징으로 하는 타겟 서버 제어 방법이 제공된다.

또한, 타겟 시스템(Target System)에서 운영되는 내장형 실시간 소프트웨어(Embedded Real-Time Software)를 호스트 시스템(Host System) 상의 개발 도구를 이용하여 원격 개발할 수 있도록 하는 타겟 서버(Target Server) 장치가 호스트 시스템 상의 도구에 대한 서비스를 하기 위하여 서비스 데몬(Service Demon)으로 작동하는 방법에 있어서, 상기 타겟 시스템에 대한 여러 가지 설정 정보를 입력하고, 상기 호스트 시스템의 타겟 서버와 통신을 담당하는 기능을 수행하는 타겟 에이전트(Target Agent)의 통신 방식에 따라, 통신 방식을 선택하는 제 1 단계와; 상기 제 1 단계에서 입력된 설정 정보 및 상기 호스트 시스템 상의 개발 도구에서 입력된 정보에 따라 변환하고자 하는 실행 파일 형식(OMF: Object Module Format)을 선택한 후, 상기 타겟 시스템의 운영 체제 파일을 선택하고, 상기 타겟 서버의 작동 시작 요청을 하는 제 2 단계와; 상기 제 2 단계에서의 작동 시작 요청이 있으면, 상기 타겟 서버와 상기 타겟 에이전트와의 연결을 수행하여, 필요한 정보를 상기 타겟 서버에 전송한 후, 상기 타겟 시스템의 운영 체제 실행 파일을 읽어서, 타겟 시스템 심볼 테이블을 작성하고, 상기 타겟 서버를 초기화하여 타겟 서버 데몬을 구동하는 제 3 단계를 포함하여 이루어진 것을 특징으로 하는 타겟 서버 장치의 서비스 데몬 작동 방법이 제공된다.

또한, 타겟 서버(Target Server) 장치를 이용하여, 호스트 시스템 상에서 교차 컴파일(Cross Compile)한 내장형 소프트웨어를 타겟 시스템에 로딩하여 원격 개발하는 방법에 있어서, 상기 호스트 시스템 상의 개발 도구들이 서비스를 요청하면, 실행 파일 형식에 따라 해독할 수 있는 정보 및 해당 실행 파일에 관련된 심볼(Symbol)들을 추출하는 제 1 단계와; 상기 제 1 단계에서 추출한 정보 및 심볼에 관련된 텍스트 및 데이터 등의 세그먼트(Segment) 정보를 추출하여, 내장형 소프트웨어를 로딩하는 제 2 단계와; 상기 제 2 단계에서 로딩된 내장형 소프트웨어를 실행하여 정확성 검사를 수행하는 제 3 단계를 포함하여 이루어진 것을 특징으로 하는 타겟 서버 장치를 이용한 내장형 소프트웨어의 원격 개발 방법이 제공된다.

또한, 타겟 시스템(Target System)에서 운영되는 내장형 실시간 소프트웨어(Embedded Real-Time Software)를 호스트 시스템(Host System) 상의 개발 도구를 이용하여 원격 개발할 수 있도록 하는 타겟 서버(Target Server)와 상기 호스트 시스템 상의 개발 도구들과의 통신을 위하여, 상기 호스트 시스템 상의 개발 도구와 상기 타겟 서버와의 연결, 비연결 상태를 나타내거나, 상기 타겟 서버 데몬 및 상기 타겟 시스템을 재시작하거나, 상기 타겟 서버의 리스트를 확인하기 위하여 입력, 수정, 조회 및 삭제할 수 있는 세션 관리 영역과; 상기 타겟 시스템에 로딩한 실행 파일을 수행하거나, 상기 호스트의 타겟 서버와 내장형 소프트웨어의 실시간 커널과의 통신을 담당할 타겟 에이전트에 관한 유형을 세팅하기 위하여 입

력, 수정, 조회 및 삭제할 수 있는 타겟 정보 및 타겟 오퍼레이션 영역과; 상기 타겟 시스템 심볼 정보 및 로딩할 실행 파일에 관련된 심볼의 테이블 관리를 위하여 입력, 수정, 조회 및 삭제할 수 있는 심볼 관리 영역과; 상기 타겟 시스템의 메모리의 일부를 필요에 따라 쓰기 및 읽기를 수행하기 위하여 입력, 수정, 조회 및 삭제할 수 있는 메모리 관련 영역과; 컨텍스트(Context)를 유지하면서, 디버깅 기능을 수행하기 위하여 입력, 수정, 조회 및 삭제할 수 있는 컨텍스트 관리 및 디버깅 영역과; 상기 호스트 시스템 상의 개발 도구, 상기 타겟 서버 장치 및 상기 타겟 에이전트간에 주고 받을 수 있는 이벤트 관리를 위하여 입력, 수정, 조회 및 삭제할 수 있는 이벤트 관리 영역을 포함하여 이루어진 것을 특징으로 하는 타겟 서버(Target Server)와 상기 호스트 시스템 상의 개발 도구들과의 통신 데이터를 기록한 컴퓨터로 읽을 수 있는 기록 매체가 제공된다.

또한, 타겟 시스템(Target System)에서 운영되는 내장형 실시간 소프트웨어(Embedded Real-Time Software)를 호스트 시스템(Host System) 상의 개발 도구를 이용하여 원격 개발할 수 있도록 하는 타겟 서버(Target Server) 제어 프로그램을 실행시킬 수 있는 컴퓨터로 읽을 수 있는 기록 매체에 있어서, 기 설정한 플래그(Flag)를 반영하여 그 특성에 맞게 상기 타겟 서버를 구동시키고, 상기 호스트 시스템 상의 도구가 요청한 기능을 해석하여 거기에 맞는 서비스를 호출해 주는 제 1 단계와; 상기 제 1 단계에서 호출된 서비스에 따라, 상기 타겟 시스템의 시스템 심볼 테이블과 상기 타겟 시스템에 로딩된 모든 실행 모듈들의 서브루틴, 변수 및 모듈 id에 대한 정보를 관리하여 상기 호스트 시스템에 있는 실행 파일을 상기 타겟 시스템에 로딩/언로딩한 후, 로딩된 모듈 리스트를 상기 호스트 시스템 상에서 관리하는 제 2 단계를 포함하여 이루어진 것을 실행시킬 수 있는 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체가 제공된다.

또한, 타겟 시스템(Target System)에서 운영되는 내장형 실시간 소프트웨어(Embedded Real-Time Software)를 호스트 시스템(Host System) 상의 개발 도구를 이용하여 원격 개발할 수 있도록 하는 타겟 서버(Target Server) 장치가 호스트 시스템 상의 도구에 대한 서비스를 하기 위하여 서비스 데몬(Service Demon)으로 작동하는 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체에 있어서, 상기 타겟 시스템에 대한 여러 가지 설정 정보를 입력하고, 상기 호스트 시스템의 타겟 서버와 통신을 담당하는 기능을 수행하는 타겟 에이전트(Target Agent)의 통신 방식에 따라, 통신 방식을 선택하는 제 1 단계와; 상기 제 1 단계에서 입력된 설정 정보 및 상기 호스트 시스템 상의 개발 도구에서 입력된 정보에 따라 변환하고자 하는 실행 파일 형식(OMF: Object Module Format)을 선택한 후, 상기 타겟 시스템의 운영 체제 파일을 선택하고, 상기 타겟 서버의 작동 시작 요청을 하는 제 2 단계와; 상기 제 2 단계에서의 작동 시작 요청이 있으면, 상기 타겟 서버와 상기 타겟 에이전트와의 연결을 수행하여, 필요한 정보를 상기 타겟 서버에 전송한 후, 상기 타겟 시스템의 운영 체제 실행 파일을 읽어서, 타겟 시스템 심볼 테이블을 작성하고, 상기 타겟 서버를 초기화하여 타겟 서버 데몬을 구동하는 제 3 단계를 포함하여 이루어진 것을 실행시킬 수 있는 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체가 제공된다.

또한, 타겟 서버(Target Server) 장치를 이용하여, 호스트 시스템 상에서 교차 컴파일(Cross Compile)한 내장형 소프트웨어를 타겟 시스템에 로딩하여 원격 개발하는 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체에 있어서, 상기 호스트 시스템 상의 개발 도구들이 서비스를 요청하면, 실행 파일 형식에 따라 해독할 수 있는 정보 및 해당 실행 파일에 관련된 심볼(Symbol)들을 추출하는 제 1 단계와; 상기 제 1 단계에서 추출한 정보 및 심볼에 관련된 텍스트 및 데이터 등의 세그먼트(Segment) 정보를 추출하여, 내장형 소프트웨어를 로딩하는 제 2 단계와; 상기 제 2 단계에서 로딩된 내장형 소프트웨어를 실행하여 정확성 검사를 수행하는 제 3 단계를 포함하여 이루어진 것을 실행시킬 수 있는 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체가 제공된다.

아래에서, 본 발명에 따른 양호한 일 실시예를 첨부한 도면을 참조로 하여 상세히 설명하겠다.

도 2는 본 발명에 일 실시예에 따른 내장형 소프트웨어의 원격 개발 도구와 타겟 시스템의 연결 구조도로서, 이를 상세히 설명하면 다음과 같다.

종래의 내장형 소프트웨어의 원격 개발 도구와는 달리 본 발명의 일 실시예에 따르면, 도구가 타겟 시스템에 연결해서 원하는 기능을 수행하기 위해서는 필요한 기능과 데이터 등의 공통 요소를 모아서 도구와 타겟 시스템 사이의 도구 중개자가 수행하도록 하는 구조를 고안하였다. 즉, 타겟 서버 장치는 도 2의 120에서 도구 중개자에 해당한다. 이와 같은 구조를 지원하기 위해 타겟 서버는 호스트 시스템 상의 도구들과 타겟 시스템 사이에서 중개하는 역할을 한다. 이러한 구조를 채택하면서 얻을 수 있는 이익은 하나의 타겟 시스템에 하나의 채널만을 유지함으로써, 타겟 시스템의 제한된 자원을 절약하는 효과를 얻을 수 있고, 도구가 타겟 시스템에 접근할 수 있는 공통 인터페이스 함수를 정의함으로써, 현재의 개발 환경에 새로운 도구를 추가하는 것이 용이해진다.

또한, 타겟 서버 장치가 타겟 시스템 심볼 테이블과 같이 변하지 않는 타겟 시스템 정보를 유지하여 굳이 타겟과 통신하지 않고도 도구가 원하는 정보를 호스트 상의 타겟 서버 장치가 서비스해줄 수 있는 장점이 있다.

도 3은 본 발명의 일 실시예에 따른 내장형 소프트웨어의 원격 개발 시스템의 구성도로서, 이를 상세히 설명하면 다음과 같다.

본 발명이 적용되는 환경은 호스트 시스템(210)과 타겟 시스템(220)이 시리얼(Serial) 혹은 이더넷(Ethernet) 방식으로 연결되어 있는 있는 내장 프로세서의 실시간 운영 체제 위에 탑재할 응용 프로그램을 개발할 수 있는 시스템(200)에서이다. 상기 호스트 시스템(210)은 하드웨어인 호스트 보드(214), 개발 지원 도구가 동작할 수 있는 호스트 운영 체제(213, 본 발명에서는 윈도우 NT로 설계하였다.)와 본 발명과 가장 관련이 깊은 타겟 서버(212)가 있고 그 위에 원격 디버거, 대화형 쉘, 자원 모니터와 같은 여러 개발 도구들(211)이 있다. 상기 타겟 시스템(220)은 디지털 텔레비전 등 가전 제품이나 이동 단말기에 내장될 수 있는데, 내장형 프로세서에 해당하는 타겟 보드(225)가 있고 내장형 실시간 커널(224)과 상기 호스트 시스템(210)의 타겟 서버(212)와 통신을 담당할 타겟 에이전트(223)와 그래픽 라이브러리와 같은 실시간 응용 라이브러리(222)가 있고, 그 위에 개발하고자 하는 응용 프로그램(221)이 탑재될 것이다. 소스 레벨 디버깅 등을 통해 상기 타겟 시스템(220)에 탑재될 응용 개발이 끝나면 상기

타겟 에이전트(223)는 그 역할을 다하게 된다.

즉, 호스트 시스템 상의 도구들(211)과 타겟 서버(212)와 타겟 에이전트(223)는 타겟 시스템(220)에 탑재할 응용 개발을 위해 필요하다. 타겟 서버(212)는 본 발명에서 다루게 될 주요 요소로서 호스트 시스템 상의 도구들(211)이 타겟 시스템(220)에 접근하기 위해서는 반드시 상기 타겟 서버(212)를 거쳐야 한다.

도 4는 본 발명의 일 실시예에 따른 타겟 서버에 대한 구조도로서, 상기 타겟 서버(300)는 타겟 서버 컨트롤기(310), 도구 인터페이스 프로토콜(320), 통신 벡앤드 관리기(330), 벡엔드 프로토콜(340), 심볼 테이블 관리기(350), 실행 모듈 관리기(360) 및 타겟 메모리 관리기(370)로 구성되어 있다.

이를 상세히 설명하면 다음과 같다.

상기 타겟 서버 컨트롤기(310)는 타겟 서버 내부에서 일어나는 모든 것을 총괄하는 지휘 본부 역할을 한다. 응용 개발자가 설정한 플래그(Flag)를 반영하여 그 특성에 맞게 상기 타겟 서버(300)를 구동시키고, 호스트 시스템 상의 도구가 요청한 기능을 해석하여 거기에 맞는 서비스를 호출해 주는 기능을 담당한다. 다시 말해 도구의 사용자가 사용하는 통신 방식, 실행 모듈 형식 등에 따라 적절한 모듈들을 구동시켜 하나의 타겟 서버 서비스 데몬(Demon)을 만들어서 도구의 요구를 기다리는 형태가 되도록 하는 모듈이다.

상기 도구 인터페이스 프로토콜(320)은 호스트 시스템 상의 도구들(211)에게 공통적인 인터페이스를 제공하여 타겟 시스템과 연결 창구를 일원화하기 위함이다. 상기 프로토콜은 타겟 서버 연결 기능, 심볼 처리 기능, 실행 시간 타겟 시스템 정보 제공 기능, 이벤트 관리 기능, 실행 모듈 로딩 기능, 디버깅 지원 기능, 메모리 관련 기능, 타겟 함수 호출 기능 등 도구가 상기 타겟 서버(300)에게 요청할 수 있는 모든 기능을 함수별로 정의하여 하나의 API(응용 프로그램 인터페이스 : Application Program Interface), 혹은 여러 개의 APIs로 원하는 기능을 실현할 수 있다. 상기 APIs는 통신을 통해 도구들로부터 요청을 받으면, 상기 타겟 서버 컨트롤기(310)가 해당 API에 대한 서비스 함수를 호출해준다.

상기 심볼 테이블 관리기(350)는 타겟 시스템의 시스템 심볼 테이블과 타겟 시스템에 로딩된 모든 실행 모듈들의 서브루틴, 변수 및 모듈 id에 대한 정보를 관리한다. 원격 디버거와 대화형 쉘과 같은 호스트 시스템 상의 도구들(211)이 서브루틴을 호출하는 것은 상기 심볼 테이블 관리자(350)를 통해서 가능하다. 특히, 로드된 모듈에 대한 심볼릭 디스어셈블리(Symbolic Disassembly) 및 특정 태스크의 서브루틴 호출을 심볼릭하게 추적할 필요가 있는 소스 레벨 원격 디버깅에 유용하다.

상기 실행 모듈 관리기(360)는 크게 호스트 시스템에 있는 실행 파일을 타겟 시스템에 로딩/언로딩하는 기능, 로딩된 모듈 리스트를 호스트 시스템 상에서 관리하는 기능을 지원한다. 또한, 상기 실행 모듈 관리기(360)는 로더(Loader)를 포함하는데 상기 로더는 여러 가지 파일 형식(예를 들어 COFF, ELF, a.out 등)을 지원하기에 용이한 DLL(Dynamic Linked Library) 구조를 갖고 있다. 상기 타겟 서버(300) 구동 시에 해당 파일 형식을 지원해주는 DLL이 타겟 서버 데몬에 연결된다.

상기 타겟 메모리 관리기(370)는 도구들이 내장형 응용을 로딩 요구할 때, 타겟 시스템의 메모리의 할당을 요구할 수 있다. 실시간 운영 체제 구동 시에 도구들이 사용할 수 있는 타겟 메모리 크기를 운영 체제 설정 파일에 세팅할 수 있고, 상기 타겟 메모리 관리자(370)는 처음에는 그 크기만 관리하다가 다 쓰게 되면 타겟 시스템에 추가 요청하여 관리해 준다. 특히, 타겟 억세스(Access) 횟수를 줄이기 위해 논리적인 캐쉬(Cache)를 두어 메모리 관리기에서 관리할 수도 있다. 즉, 상기 실행 모듈 관리기(360)의 로더가 재배치를 수행할 때, 이 캐쉬를 일종의 버퍼처럼 사용하여 재배치가 끝났을 때, 한번만 타겟 메모리에 쓰기를 수행하기도 하고, 캐쉬 블록 별로 속성을 부여하여 실행 모듈의 텍스트 섹션 같은 경우는 타겟 시스템과 똑같은 메모리 카피를 호스트 시스템 상의 이 캐쉬에 둠으로써 타겟 시스템 접근 횟수를 감소시킨다.

상기 통신 벡엔드 관리기(330)는 벡엔드 관리기 내 공통 서비스 루틴, 이더넷, 시리얼과 같은 통신을 지원해주는 각각의 벡엔드들로 이루어져 있다. 각각의 벡엔드들은 DLL로 이루어져 있어 타겟 서버 구동 시 설정에 따라 해당 벡엔드가 타겟 서버 데몬에 링크된다.

상기 벡엔드 프로토콜(340)은 타겟 에이전트와 통신하기 위하여 필요하다. 상기 프로토콜은 이더넷, 시리얼 통신에 공통적으로 적용된다. 상기 벡엔드 프로토콜(340)은 상기 도구 인터페이스용 프로토콜(320) 중 타겟에 접속이 필요한 APIs를 기능에 따라 몇 가지의 프로토콜로 정의하여 호스트 상의 벡엔드와 타겟 에이전트 사이에 Call/Return이 반복되게 된다.

도 5는 도 4에 도시된 도구 인터페이스 프로토콜의 기능에 따른 분류 테이블로서, 이를 상세히 설명하면 다음과 같다.

상기 도구 인터페이스 프로토콜은 호스트 시스템 상의 모든 도구들이 사용할 수 있도록 일반화 시킨 것이다. 따라서, 현재의 개발 환경에 새로운 도구를 추가하는 경우에도 타겟 서버 내부에서 처리하는 방식에 상관없이 상기 프로토콜만 사용하면 된다. 또한 상기 프로토콜은 타겟 에이전트와 접속하기 위해 타겟 서버 장치가 필요한 처리를 수행한 뒤, 필요에 따라 벡엔드 프로토콜을 호출하여 그 결과를 도구에 반환한다. 즉, 상기 프로토콜의 일부는 벡엔드 프로토콜에도 존재하게 되는 것이다. 상기 프로토콜에 대하여 자세히 설명하면 다음과 같다.

상기 호스트 시스템 상의 도구와 타겟 서버 장치 간의 인터페이스 프로토콜은 크게 도구가 타겟 서버와 연결/비연결하거나 타겟 서버 데몬 및 타겟 시스템을 재시작, 타겟 서버 리스트를 확인하는 것과 같은 세션 관리 기능 프로토콜과, 타겟 시스템에 로딩한 실행 파일을 수행하거나 타겟 에이전트 타입을 세팅하기 위한 타겟 정보 및 타겟 오퍼레이션 수행 기능 프로토콜과, 타겟 시스템 심볼 정보 및 로딩할 실행 파일에 관련된 심볼을 추가 삭제하기 위한 심볼 테이블 관리를 위한 프로토콜과, 타겟 메모리의 일부를 타겟 서버 장치의 메모리 관리자가 관리하면서 필요에 따라 타겟의 메모리에 쓰기/읽기 등의 수행할 수 있도록 하기 위한 메모리 관련 프로토콜과, 원격 디버깅 등을 수행하기 위해 컨텍스트를 유지하면서 디버깅 기능을 수행할 수 있는 컨텍스트 관리 및 디버깅 관련 프로토콜과, 도구와 타겟 서버 장치와 타겟 에이전트 간에 주고 받을 수 있는 이벤트 관리 기능을 수행하기 위한 프로토콜로 분류되고, 이들 프로토콜은 세부적으로 총 54 개이고 각각의 프로토콜을 메시지 번호로 정의하여 도구의 서비스 호출 시 타겟 서버 컨트

롤기가 서비스를 호출할 수 있는 서비스 번호로 사용한다.

도 6은 본 발명의 일 실시예에 따른 타겟 서버 장치가 도구에 대한 서비스를 제공하기 위하여 서비스 데몬으로 동작되는 과정을 나타내는 흐름도로서, 이를 상세히 설명하면 다음과 같다.

먼저, 스텝 S510에서, 타겟 서버 데몬을 구동시키기 전에 사용하려는 타겟 시스템에 대한 여러 가지 설정 정보를 입력해야 한다. 이는 상기 타겟 서버 컨트롤기(310)가 그래픽 사용자 인터페이스로 실현되면 사용자가 사용하기 편리하기 때문이다. 먼저 타겟 서버(212)와 타겟 에이전트(223)의 통신 방식이 어떤 방식으로 연결될 것인지 선택한다. 예를 들어, 이더넷 방식이라면 타겟 IP 주소, 타임 아웃 등의 정보를 주고, 시리얼 방식이라면 전송 속도(bps), 포트 번호, 타임 아웃 등의 정보를 준다.

이어서, 스텝 S520에서, 여러 가지 실행 파일 형식(OMF: Object Module Format)을 선택하는데 이는 타겟 서버 장치가 여러 실행 파일 형식을 지원할 수 있는 구조로 실현될 수 있기 때문이다. 즉, 상기 실행 모듈 관리기(360)에서 타겟 시스템에 로딩하는데 필요한 공통 로딩 기능은 로더에 두고, 각 OMF 리더(Reader)는 각각 DLL로 실현하여 타겟 서버 장치 구동 시 링크될 수 있게 한 것이다.

이어서, 스텝 S530에서, 타겟 시스템 심볼 테이블을 타겟 서버 장치가 유지하기 위해 타겟 시스템에 탑재된 실시간 운영 체제 실행 파일을 선택한 후, 스텝 S540에서, 하나의 호스트 시스템에는 여러 개의 타겟 시스템을 연결할 수도 있으므로 한 타겟 시스템 당 하나의 타겟 서버 장치 즉, 여러 개의 타겟 서버 장치가 데몬 형식으로 동작할 수 있으므로 이들 간의 구분자로서 타겟 서버 이름을 부여할 수 있도록 실현한다. 이렇게 타겟 서버 장치를 구동하기 위한 설정이 끝나면, 스텝 S550에서, 타겟 서버 시작 요청을 상기 타겟 서버 컨트롤기(310)에게 한다.

이어서, 스텝 S560에서, 같은 이름, 혹은 같은 IP의 타겟 서버가 존재하는지 판단한다.

상기 스텝 S560에서의 판단 결과, 타겟 서버가 존재하면, 상기 스텝 S510으로 복귀하고, 존재하지 아니하면, 스텝 S570에서, 설정한 통신 백엔드.dll을 상기 통신 백엔드 관리기(330)를 통해 구동한 후, 스텝 S580에서, 타겟 서버 장치의 타겟 시스템에서의 대응 요소인 상기 타겟 에이전트(223)와 연결을 시도하고, 타겟 시스템에 필요한 정보, 예를 들어, 도구가 사용하는 타겟 메모리 시작 주소 및 크기 등을 반환받아 타겟 서버 초기화에 사용한다.

이어서, 스텝 S590에서, 해당 OMF.dll을 구동하고, 스텝 S592에서, 타겟 운영 체제 실행 파일을 읽어서, 운영 체제 파일을 코딩한다.

마지막으로, 스텝 S594에서, 에러 처리 등 타겟 서버 내부 모듈들을 초기화한 후, 스텝 S596에서, 타겟 서버 데몬이 구동된다.

도 7은 본 발명의 일 실시예에 따른 타겟 서버 장치가 데몬으로 작동하고 있으면서, 호스트 상의 여러 개발 도구와 타겟 에이전트(223)의 서비스 요청을 받아들여 서비스해 주는 과정을 나타낸 흐름도로서, 이를 상세히 설명하면 다음과 같다.

먼저, 스텝 S610에서, 타겟 서버 데몬에 대한 종료 요청이 있는지를 판단하여, 종료 요청이 있으면, 데몬이 종료되고, 종료 요청이 없으면, 스텝 S620에서, 호스트 시스템 상의 도구(211)나 타겟 에이전트(223)의 이벤트에 대하여 대기 상태로 설정된다.

임의의 도구가 타겟 서버의 서비스를 받기 위해 최초로 사용하는 프로토콜은 도 5의 TOOL_ATTACH인데, 타겟 서버 데몬은 상기 프로토콜을 받으면 도구들을 구별할 수 있는 구분자를 배당하게 되고, 이를 이용하여 이하 다른 프로토콜을 사용할 수 있도록 되어 있으므로, 스텝 S630에서, 타겟 서버는 임의의 도구가 최초로 요청하는 서비스를 판단한다.

상기 스텝 S630에서의 판단 결과, 최초의 요청이면, 스텝 S640에서, 해당 서비스를 찾아서 관련된 프로세스를 수행한 후, 스텝 S642에서, 다른 도구의 서비스 요청과 구분하기 위한 구분자를 부여해 준 후, 상기 스텝 S620의 복귀한다.

상기 스텝 S630에서의 판단 결과, 최초 요청이 아니면, 스텝 S650에서, 타겟 시스템에서 도착한 이벤트인지, 호스트 시스템 상의 도구(211)가 타겟 서버 데몬과 연결되고 난 이후의 서비스 요청인지를 판단한다.

상기 스텝 S650에서의 판단 결과, 타겟 시스템에서 도착한 이벤트이면, 스텝 S660에서, 해당 도구에게 전달하여 준 후, 상기 스텝 S620으로 복귀하고, 도구의 요청이면, 스텝 S670에서, 해당 서비스를 찾은 후, 스텝 S672에서, 서비스 요청 인자로 받은 도구의 구분자를 확인하여, 스텝 S674에서, 해당 도구에게 원하는 서비스를 제공한 후, 상기 스텝 S620으로 복귀한다.

도 7에서 보는 바와 같이, 타겟 서버 데몬은 도구와 타겟의 이벤트를 받아 처리하고 나면 다음 서비스 요청을 위해 항상 대기하고 있다.

도 8은 본 발명의 일 실시예에 다른 타겟 서버 장치 내의 각 요소 간의 관계를 표현하기 위해 호스트에서 교차 컴파일(Cross Compile)한 내장형 소프트웨어를 타겟 서버 장치를 이용하여 타겟에 로딩하여 실행하는 과정을 나타낸 흐름도로서, 이를 상세히 설명하면 다음과 같다.

먼저, 교차 컴파일이 끝난 후, 원격 디버거나 대화형 쉘과 같은 도구들은 이 실행 파일을 타겟 시스템에 로딩하여 호스트 시스템 상에서 시험하면서 프로그램의 오류를 찾아가게 된다. 이를 위해서 호스트 개발 도구들(211)은 스텝 S710에서, 실행 파일 로딩에 관련된 프로토콜(OBJ_MODULE_LOAD) 사용 서비스를 요청하게 되고, 타겟 서버 컨트롤기는 해당 API를 호출하게 된다. 타겟 서버 내부에서 이를 해결 해주는 모듈은 도 4에 있는 상기 실행 모듈 관리기(360)의 로더와 실행 파일 형식에 따라 읽어들일 수 있는 리더이다.

이어서, 스텝 S720에서, 로더가 실행 파일 형식에 따라 실행 파일을 해독할 수 있는 리더에게 실행 파일에 대한 정보를 요청하면, 스텝 S730에서, 리더는 헤더를 읽어서 해당 파일에 대한 정보를 로더에게 반환한 후, 스텝 S740에서, 리더는 해당 실행 파일 관련된 심볼들을 추가해줄 것을 심볼 테이블 관리기에게

요청한다.

이어서, 스텝 S750에서, 상기 심볼 테이블 관리기(350)는 심볼을 추가하는데, 이 때 심볼은 여러 타입으로 분류되고, 다른 실행 파일과 관련된 심볼과 구분하기 위해 ID를 저장하도록 설계하였다. 상기 심볼 테이블 관리기(350)가 해당 심볼들을 저장하고 난 후에, 스텝 S760에서, 로더는 리더에게 텍스트, 데이터 등의 세그먼트(Segment) 정보를 요청한다.

이어서, 스텝 S770에서, 리더는 세그먼트 정보를 반환하고 상기 타겟 메모리 관리기(370)에게 세그먼트별로 캐쉬에 저장해 줄 것을 요청하면, 스텝 S780에서, 재배치의 필요 유무를 판단하여, 재배치가 필요하면, 스텝 S790에서, 로더가 재배치를 수행하고, 그러하지 아니하면, 스텝 S792에서, 상기 타겟 메모리 관리기(370)는 세그먼트 주소를 계산하여, 필요한 메모리 크기를 확보하여야 한다. 이 때, 도구가 사용하기 위해 고정적으로 할당 받은 타겟 메모리 크기가 부족하면 메모리를 추가적으로 할당 받아 상기 타겟 메모리 관리기(370)가 관리하는 메모리 풀에 추가해야 한다.

이어서, 각 세그먼트를 타겟에 쓰기 위한 메모리가 확보된 후, 스텝 S794에서, 상기 타겟 메모리 관리기(370)는 상기 통신 백엔드 관리기(330)에게 상기 타겟 메모리 관리기(370)를 통해 계산한 주소에 쓰기 요청을 한 후, 호스트 시스템에서 타겟 시스템으로 하나의 내장형 소프트웨어를 로딩하고 나면, 스텝 S796에서, 호스트 시스템 상의 도구(211)는 실행에 관련된 도구 인터페이스 프로토콜(FUNC_CALL, DIRECT_CALL)을 사용하여 실행 요청을 하며, 로딩된 소프트웨어가 타겟 시스템에서 잘 동작되는 지 확인해 볼 수 있고, 문제가 발견되면, 타겟 서버 장치가 제공하는 해당 프로토콜을 이용하여 해결해 나갈 수 있다.

상기와 같은 본 발명은 컴퓨터로 읽을 수 있는 기록 매체로 기록되고, 컴퓨터에 의해 처리될 수 있다.

발명의 효과

앞서 상세히 설명한 바와 같이 본 발명의 타겟 서버 장치는 도구와 호스트 시스템 사이의 통신, 심볼 테이블 관리, 타겟 메모리 관리, 실행 파일 관리 등을 모두 책임짐으로써, 도구들이 타겟 시스템에 접근하여 원격 개발하기 위해 필요한 기능을 도구 인터페이스 프로토콜을 사용하는 것으로 모두 해결해 주어, 새로운 도구를 추가하기가 매우 용이하다는 효과가 있다.

또한, 도구 - 타겟 시스템 간의 채널을 타겟 서버 장치가 일원화하여 관리해줌으로써, 타겟 시스템의 자원을 절약해 주는 효과가 있고, 타겟 서버 장치의 심볼 테이블 관리기, 타겟 메모리 관리기를 통해 호스트-타겟 통신 횟수를 절감시킬 수 있는 효과가 있다.

이상에서 본 발명에 대한 기술 사상을 첨부 도면과 함께 서술하였지만 이는 본 발명의 가장 양호한 일 실시예를 예시적으로 설명한 것이지 본 발명을 한정하는 것은 아니다. 또한, 이 기술 분야의 통상의 지식을 가진 자이면 누구나 본 발명의 기술 사상의 범주를 이탈하지 않는 범위 내에서 다양한 변형 및 모방이 가능함은 명백한 사실이다.

(57) 청구의 범위

청구항 1. 타겟 시스템(Target System)에서 운영되는 내장형 실시간 소프트웨어(Embedded Real-Time Software)를 호스트 시스템(Host System) 상의 개발 도구를 이용하여 원격 개발할 수 있도록 하는 타겟 서버(Target Server) 장치에 있어서,

기 설정한 플래그(Flag)를 반영하여 그 특성에 맞게 상기 타겟 서버를 구동시키고, 상기 호스트 시스템 상의 도구가 요청한 기능을 해석하여 거기에 맞는 서비스를 호출해 주는 기능을 수행하는 타겟 서버 컨트롤기와;

상기 타겟 서버 컨트롤기에 의하여 호출된 서비스에 따라, 상기 타겟 시스템의 시스템 심볼 테이블과 상기 타겟 시스템에 로딩된 모든 실행 모듈들의 서브루틴, 변수 및 모듈 id에 대한 정보를 관리하는 기능을 수행하는 심볼 테이블 관리기와;

상기 호스트 시스템에 있는 실행 파일을 타겟 시스템에 로딩/언로딩하는 기능, 로딩된 모듈 리스트를 상기 호스트 시스템 상에서 관리하는 기능을 수행하는 실행 모듈 관리기와;

타겟 서버 구동시 상기 호스트 시스템 상의 도구들과 상기 타겟 시스템이 통신할 수 있도록 하는 통신 백엔드 관리기를 포함하여 이루어진 것을 특징으로 하는 타겟 서버 제어 장치.

청구항 2. 제 1 항에 있어서,

상기 타겟 시스템의 도구가 내장형 응용을 로딩 요구하면, 사용할 메모리를 할당하여 관리하며, 상기 타겟 시스템의 억세스(Access) 횟수를 줄이기 위한 기능을 수행하는 논리적인 캐쉬부를 포함하는 타겟 메모리 관리기를 더 포함하여 이루어진 것을 특징으로 하는 타겟 서버 제어 장치.

청구항 3. 제 1 항에 있어서,

상기 실행 모듈 관리기는,

상기 타겟 시스템에 로딩된 모듈 리스트가 여러 가지 형식의 파일이면, 상기 타겟 서버 구동시에 해당되는 파일 형식을 지원하여 주는 로더부를 포함하여 이루어진 것을 특징으로 하는 타겟 서버 장치.

청구항 4. 타겟 시스템(Target System)에서 운영되는 내장형 실시간 소프트웨어(Embedded Real-Time Software)를 호스트 시스템(Host System) 상의 개발 도구를 이용하여 원격 개발할 수 있도록 하는 타겟 서버(Target Server) 제어 방법에 있어서,

기 설정한 플래그(Flag)를 반영하여 그 특성에 맞게 상기 타겟 서버를 구동시키고, 상기 호스트 시스템 상의 도구가 요청한 기능을 해석하여 거기에 맞는 서비스를 호출해 주는 제 1 단계와;

상기 제 1 단계에서 호출된 서비스에 따라, 상기 타겟 시스템의 시스템 심볼 테이블과 상기 타겟 시스템에 로딩된 모든 실행 모듈들의 서브루틴, 변수 및 모듈 id에 대한 정보를 관리하여 상기 호스트 시스템에 있는 실행 파일을 상기 타겟 시스템에 로딩/언로딩한 후, 로딩된 모듈 리스트를 상기 호스트 시스템 상에서 관리하는 제 2 단계를 포함하여 이루어진 것을 특징으로 하는 타겟 서버 제어 방법.

청구항 5. 제 4 항에 있어서,

상기 타겟 시스템의 도구가 내장형 응용을 로딩 요구하면, 사용할 메모리를 할당하며, 억세스(Access) 횟수를 줄이기 위하여 논리적인 캐쉬 기능을 수행하는 제 3 단계를 더 포함하여 이루어진 것을 특징으로 하는 타겟 서버 제어 방법.

청구항 6. 타겟 시스템(Target System)에서 운영되는 내장형 실시간 소프트웨어(Embedded Real-Time Software)를 호스트 시스템(Host System) 상의 개발 도구를 이용하여 원격 개발할 수 있도록 하는 타겟 서버(Target Server) 장치가 호스트 시스템 상의 도구에 대한 서비스를 하기 위하여 서비스 데몬(Service Demon)으로 작동하는 방법에 있어서,

상기 타겟 시스템에 대한 여러 가지 설정 정보를 입력하고, 상기 호스트 시스템의 타겟 서버와 통신을 담당하는 기능을 수행하는 타겟 에이전트(Target Agent)의 통신 방식에 따라, 통신 방식을 선택하는 제 1 단계와;

상기 제 1 단계에서 입력된 설정 정보 및 상기 호스트 시스템 상의 개발 도구에서 입력된 정보에 따라 변환하고자 하는 실행 파일 형식(OMF: Object Module Format)을 선택한 후, 상기 타겟 시스템의 운영 체제 파일을 선택하고, 상기 타겟 서버의 작동 시작 요청을 하는 제 2 단계와;

상기 제 2 단계에서의 작동 시작 요청이 있으면, 상기 타겟 서버와 상기 타겟 에이전트와의 연결을 수행하며, 필요한 정보를 상기 타겟 서버에 전송한 후, 상기 타겟 시스템의 운영 체제 실행 파일을 읽어서, 타겟 시스템 심볼 테이블을 작성하고, 상기 타겟 서버를 초기화하며 타겟 서버 데몬을 구동하는 제 3 단계를 포함하여 이루어진 것을 특징으로 하는 타겟 서버 장치의 서비스 데몬 작동 방법.

청구항 7. 제 6 항에 있어서,

상기 제 1 단계는,

상기 타겟 서버와 상기 타겟 에이전트의 통신 방식이 이더넷(Ethernet) 방식이면, 타겟 IP 주소 및 타임 아웃등의 정보를 주고, 시리얼(Serial) 방식이면, 전송 속도, 포트 번호 및 타임 아웃등의 정보를 주는 것을 특징으로 하는 타겟 서버 장치의 서비스 데몬 작동 방법.

청구항 8. 제 6 항에 있어서,

상기 제 2 단계는,

하나의 호스트 시스템에 여러 개의 타겟 시스템이 연결되어 있으면, 한 타겟 시스템 당 하나의 타겟 서버 장치가 데몬 형식으로 동작할 수 있도록 각각의 타겟 서버 이름을 부여하는 서브 단계를 포함하여 이루어진 것을 특징으로 하는 타겟 서버 장치의 서비스 데몬 작동 방법.

청구항 9. 제 6 항 또는 제 8 항에 있어서,

상기 제 3 단계는,

상기 제 2 단계에서의 작동 시작 요청이 있은 후, 같은 이름 또는 같은 IP의 타겟 서버가 존재하면, 상기 제 1 단계로 복귀하는 것을 특징으로 하는 타겟 서버 장치의 서비스 데몬 작동 방법.

청구항 10. 타겟 서버(Target Server) 장치를 이용하여, 호스트 시스템 상에서 교차 컴파일(Cross Compile)한 내장형 소프트웨어를 타겟 시스템에 로딩하여 원격 개발하는 방법에 있어서,

상기 호스트 시스템 상의 개발 도구들이 서비스를 요청하면, 실행 파일 형식에 따라 해독할 수 있는 정보 및 해당 실행 파일에 관련된 심볼(Symbol)들을 추출하는 제 1 단계와;

상기 제 1 단계에서 추출한 정보 및 심볼에 관련된 텍스트 및 데이터 등의 세그먼트(Segment) 정보를 추출하여, 내장형 소프트웨어를 로딩하는 제 2 단계와;

상기 제 2 단계에서 로딩된 내장형 소프트웨어를 실행하여 정확성 검사를 수행하는 제 3 단계를 포함하여

이루어진 것을 특징으로 하는 타겟 서버 장치를 이용한 내장형 소프트웨어의 원격 개발 방법.

청구항 11. 제 10 항에 있어서,

상기 제 1 단계는,

추출하고자 하는 심볼과 다른 실행 파일과 관련된 심볼을 구별하기 위하여 각각의 심볼에게 고유의 ID를 부여하는 서브 단계를 포함하여 이루어진 것을 특징으로 하는 타겟 서버 장치를 이용한 내장형 소프트웨어의 원격 개발 방법.

청구항 12. 제 10 항 또는 제 11 항에 있어서,

상기 제 2 단계는,

추출된 세그먼트 정보를 캐쉬에 저장한 후, 각각의 세그먼트 별로 재배치가 필요한지 여부를 판단하는 제 1 서브 단계와;

상기 제 1 서브 단계에서의 판단 결과, 재배치가 필요하면, 각각의 세그먼트 별로 재배치를 수행한 후, 각각의 세그먼트 주소를 계산하여 쓰기(Write)를 수행하고, 재배치가 필요하지 아니하면, 바로 내장형 소프트웨어를 로딩하는 제 2 서브 단계를 포함하여 이루어진 것을 특징으로 하는 타겟 서버 장치를 이용한 내장형 소프트웨어의 원격 개발 방법.

청구항 13. 타겟 시스템(Target System)에서 운영되는 내장형 실시간 소프트웨어(Embedded Real-Time Software)를 호스트 시스템(Host System) 상의 개발 도구를 이용하여 원격 개발할 수 있도록 하는 타겟 서버(Target Server)와 상기 호스트 시스템 상의 개발 도구들과의 통신을 위하여,

상기 호스트 시스템 상의 개발 도구와 상기 타겟 서버와의 연결, 비연결 상태를 나타내거나, 상기 타겟 서버 데몬 및 상기 타겟 시스템을 재시작하거나, 상기 타겟 서버의 리스트를 확인하기 위하여 입력, 수정, 조회 및 삭제할 수 있는 세션 관리 영역과;

상기 타겟 시스템에 로딩한 실행 파일을 수행하거나, 상기 호스트의 타겟 서버와 내장형 소프트웨어의 실시간 커널과의 통신을 담당할 타겟 에이전트에 관한 유형을 세팅하기 위하여 입력, 수정, 조회 및 삭제할 수 있는 타겟 정보 및 타겟 오퍼레이션 영역과;

상기 타겟 시스템 심볼 정보 및 로딩할 실행 파일에 관련된 심볼의 테이블 관리를 위하여 입력, 수정, 조회 및 삭제할 수 있는 심볼 관리 영역과;

상기 타겟 시스템의 메모리의 일부를 필요에 따라 쓰기 및 읽기를 수행하기 위하여 입력, 수정, 조회 및 삭제할 수 있는 메모리 관련 영역과;

컨텍스트(Context)를 유지하면서, 디버깅 기능을 수행하기 위하여 입력, 수정, 조회 및 삭제할 수 있는 컨텍스트 관리 및 디버깅 영역과;

상기 호스트 시스템 상의 개발 도구, 상기 타겟 서버 장치 및 상기 타겟 에이전트간에 주고 받을 수 있는 이벤트 관리를 위하여 입력, 수정, 조회 및 삭제할 수 있는 이벤트 관리 영역을 포함하여 이루어진 것을 특징으로 하는 타겟 서버(Target Server)와 상기 호스트 시스템 상의 개발 도구들과의 통신 데이터를 기록한 컴퓨터로 읽을 수 있는 기록 매체.

청구항 14. 타겟 시스템(Target System)에서 운영되는 내장형 실시간 소프트웨어(Embedded Real-Time Software)를 호스트 시스템(Host System) 상의 개발 도구를 이용하여 원격 개발할 수 있도록 하는 타겟 서버(Target Server) 제어 프로그램을 실행시킬 수 있는 컴퓨터로 읽을 수 있는 기록 매체에 있어서,

기 설정한 플래그(Flag)를 반영하여 그 특성에 맞게 상기 타겟 서버를 구동시키고, 상기 호스트 시스템 상의 도구가 요청한 기능을 해석하여 거기에 맞는 서비스를 호출해 주는 제 1 단계와;

상기 제 1 단계에서 호출된 서비스에 따라, 상기 타겟 시스템의 시스템 심볼 테이블과 상기 타겟 시스템에 로딩된 모든 실행 모듈들의 서브루틴, 변수 및 모듈 id에 대한 정보를 관리하여 상기 호스트 시스템에 있는 실행 파일을 상기 타겟 시스템에 로딩/언로딩한 후, 로딩된 모듈 리스트를 상기 호스트 시스템 상에서 관리하는 제 2 단계를 포함하여 이루어진 것을 실행시킬 수 있는 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체.

청구항 15. 타겟 시스템(Target System)에서 운영되는 내장형 실시간 소프트웨어(Embedded Real-Time Software)를 호스트 시스템(Host System) 상의 개발 도구를 이용하여 원격 개발할 수 있도록 하는 타겟 서버(Target Server) 장치가 호스트 시스템 상의 도구에 대한 서비스를 하기 위하여 서비스 데몬(Service Demon)으로 작동하는 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체에 있어서,

상기 타겟 시스템에 대한 여러 가지 설정 정보를 입력하고, 상기 호스트 시스템의 타겟 서버와 통신을 담당하는 기능을 수행하는 타겟 에이전트(Target Agent)의 통신 방식에 따라, 통신 방식을 선택하는 제 1 단계와;

상기 제 1 단계에서 입력된 설정 정보 및 상기 호스트 시스템 상의 개발 도구에서 입력된 정보에 따라 변환하고자 하는 실행 파일 형식(OMF: Object Module Format)을 선택한 후, 상기 타겟 시스템의 운영 체제

파일을 선택하고, 상기 타겟 서버의 작동 시작 요청을 하는 제 2 단계와;

상기 제 2 단계에서의 작동 시작 요청이 있으면, 상기 타겟 서버와 상기 타겟 에이전트와의 연결을 수행하여, 필요한 정보를 상기 타겟 서버에 전송한 후, 상기 타겟 시스템의 운영 체제 실행 파일을 읽어서, 타겟 시스템 심볼 테이블을 작성하고, 상기 타겟 서버를 초기화하여 타겟 서버 데몬을 구동하는 제 3 단계를 포함하여 이루어진 것을 실행시킬 수 있는 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체.

청구항 16. 타겟 서버(Target Server) 장치를 이용하여, 호스트 시스템 상에서 교차 컴파일(Cross Compile)한 내장형 소프트웨어를 타겟 시스템에 로딩하여 원격 개발하는 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체에 있어서,

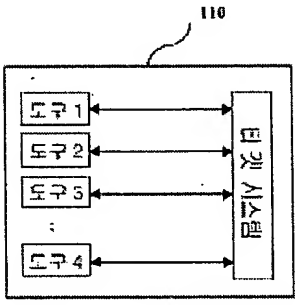
상기 호스트 시스템 상의 개발 도구들이 서비스를 요청하면, 실행 파일 형식에 따라 해독할 수 있는 정보 및 해당 실행 파일에 관련된 심볼(Symbol)들을 추출하는 제 1 단계와;

상기 제 1 단계에서 추출한 정보 및 심볼에 관련된 텍스트 및 데이터 등의 세그먼트(Segment) 정보를 추출하여, 내장형 소프트웨어를 로딩하는 제 2 단계와;

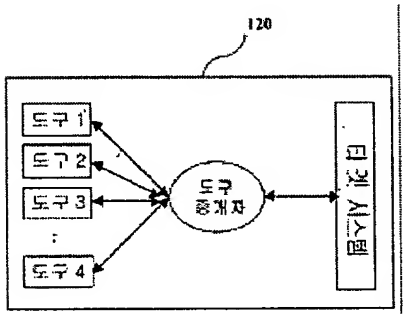
상기 제 2 단계에서 로딩된 내장형 소프트웨어를 실행하여 정확성 검사를 수행하는 제 3 단계를 포함하여 이루어진 것을 실행시킬 수 있는 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체.

도면

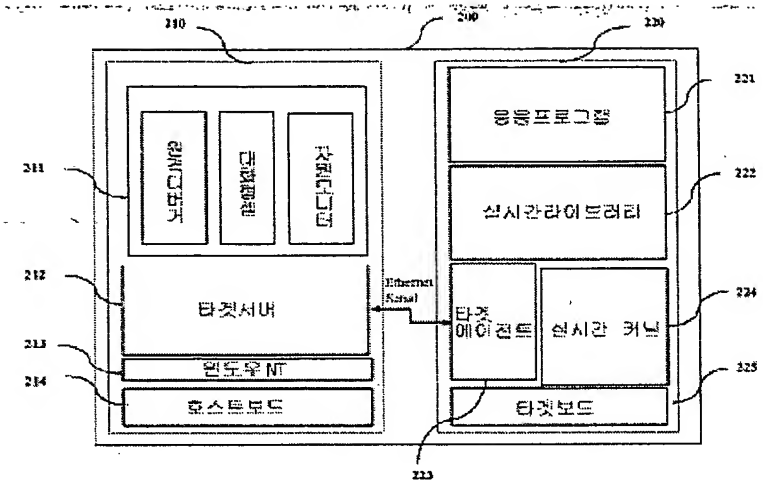
도면1



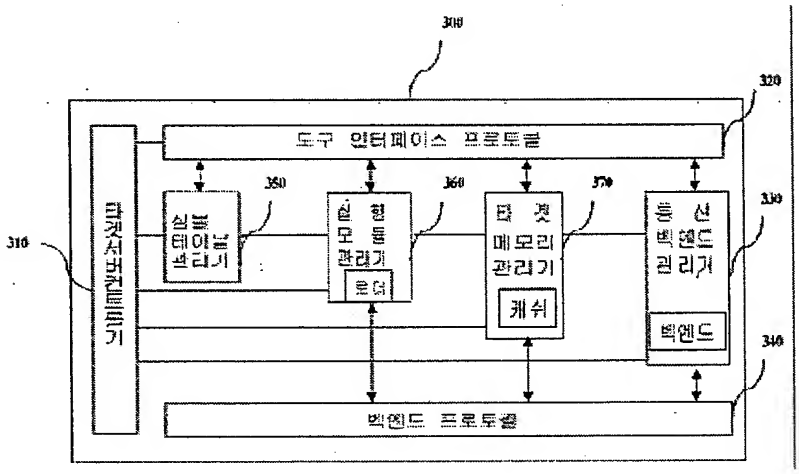
도면2



도면3


210
200
220
221
222
224
225
211
212
213
214
223
응용프로그램
실시간라이브러리
타겟서버
Ethernet
Kernel
타겟
에이전트
실시간 커널
윈도우NT
호스트보드
타겟보드

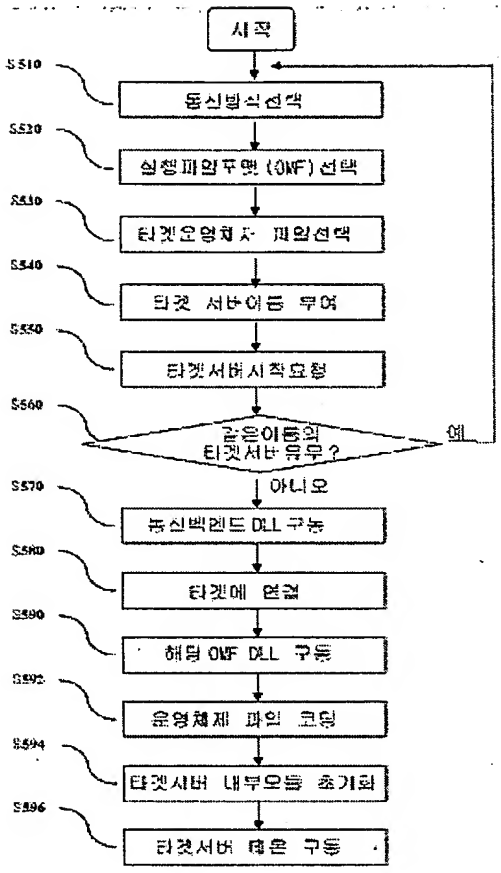
도면4


300
320
310
350
360
370
330
340
도구 인터페이스 프로토콜
심볼
테이블
관리기
실 행
모 듈
관리기
로더
타 겟
메모리
관리기
캐쉬
통 신
백엔드
관 리 기
백엔드
타겟서버컨트롤기
백엔드 프로토콜

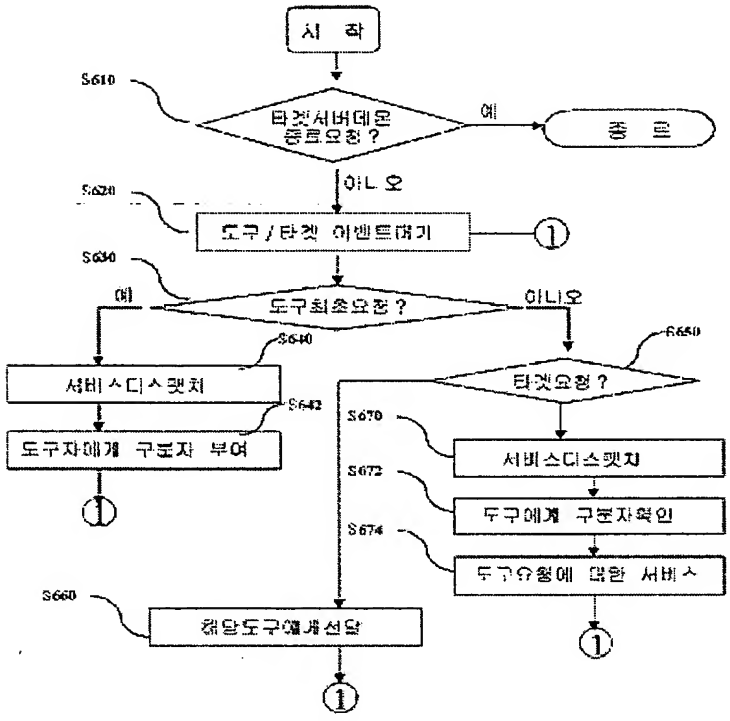
도면5

기능분류	도구인터페이스프로토콜	메시지
세션관리	TOOL_ATTACH TOOL_DETACH TS_INFO_GET TARGET_RESET INFO_SET INFO_Q_GET	1 2 3 4 5 6
타겟정보 및 타겟 오퍼레이션	FUNC_CALL SERVICE_ADD AGENT_MODE_SET AGENT_MODE_GET DIRECT_CALL	7 8 9 10 11
심볼관리	SYM_LIST_GET SYM_TBL_INFO_GET SYM_FIND SYM_ADD SYM_REMOVE	12 13 14 15 16
실행파일 모듈관리	OBJ_MODULE_LOAD OBJ_MODULE_UNLOAD OBJ_MODULE_LIST OBJ_MODULE_INFO_GET OBJ_MODULE_FIND	17 18 19 20 21
메모리관련	MEM_CHECKSUM MEM_READ MEM_WRITE MEM_SET MEM_SCAN MEM_MOVE MEM_ALLOC MEM_FREE MEM_INFO_GET MEM_ALIGN MEM_REALLOC MEM_ADD_TO_POOL REGS_GET REGS_SET	22 23 24 25 26 27 28 29 30 31 32 33 34 35
컨텍스트관리 및 디버깅	CONTEXT_CREATE CONTEXT_KILL CONTEXT_SUSPEND CONTEXT_CONT CONTEXT_RESUME CONTEXT_STEP	36 37 38 39 40 41
이벤트관리	EVENTPOINT_ADD EVENTPOINT_DELETE EVENTPOINT_LIST EVENT_OUT_REGISTER_FOR_EVENT UN_REGIS_FOR_EVENT EVENT_ADD	42 43 50 51 52 53 54

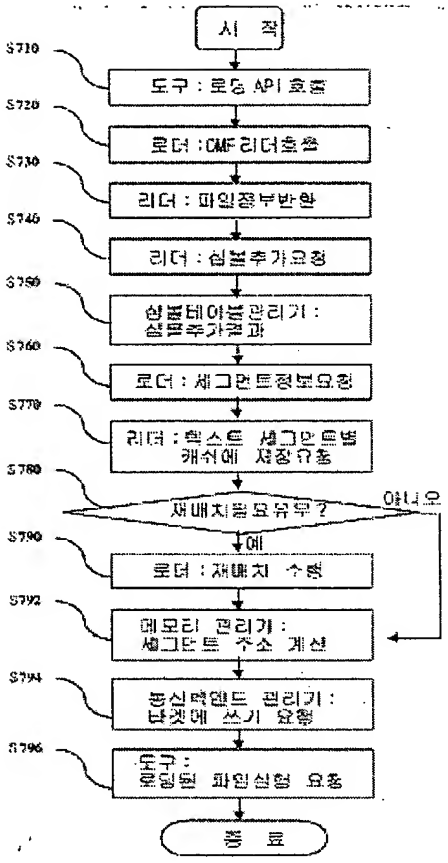
도면6


시작
S510
통신방식선택
S520
실행파일루멘(DNF)선택
S530
타겟운영체제 파일선택
S540
타겟 서버이름 부여
S550
타겟서버시작요청
S560
같은이름의 타겟서버유무?
예
아니오
S570
통신백엔드 DLL 구동
S580
타겟에 연결
S590
해당 DNF DLL 구동
S592
운영체제 파일 코딩
S594
타겟서버 내부모듈 초기화
S596
타겟서버 데몬 구동

도면7


시 작
S610
타겟서버에온 종료요청?
예
종 료
아니오
S620
도구/타겟 이벤트대기
①
S630
도구최초요청?
예
아니오
S640
서비스디스패치
S642
도구자에게 구분자 부여
①
S650
타겟요청?
S670
서비스디스패치
S672
도구에게 구분자확인
S674
도구유형에 맞는 서비스
①
S660
해당도구에게전달
①

도면8


시 작
S710
도구 : 로딩 API 호출
S720
로더 : OMF 리더호출
S730
리더 : 파일정보반환
S740
리더 : 심볼추가요청
S750
심볼테이블관리기 :
심볼추가결과
S760
로더 : 세그먼트정보요청
S770
리더 : 텍스트 세그먼트별
캐쉬에 저장요청
S780
재배치필요유무?
아니오
예
S790
로더 : 재배치 수행
S792
메모리 관리기 :
세그먼트 주소 계산
S794
동신백엔드 관리기 :
보겟에 쓰기 요청
S796
도구 :
로딩된 파일실행 요청
종 료